Examples of documentation created for British Aerospace and examples of various programming languages.

System Specification Section 1
IBM to VAX File Transfer Subroutine

Test Procedure Section 2
IBM to VAX File Transfer Subroutine

User Manual Section 3
ACES DTF File Transfer

System Manual Section 4
IBM to VAX File Transfer Subroutine

VAX DCL Section 5
Define Local Printer Command File

VAX FORTRAN Section 6
IBM to VAX File Transfer Subroutine

VAX BASIC Section 7
A CLAWS Virtual Text Function

VAX COBOL Section 8
PANOPLIE Database Update Program

MUMPS Section 9
Patch Tape Verification Routine

DOS batch Section 10
Duplicate File Finder

Turbo C Section 11
BEATES Menu Manager

The documentation and programs included in this portfolio are samples of work created by Iain White, they are copyright to the appropriate company as detailed.

1-SYSTEM SPECIFICATION

Project:	ACES DTF File Transfer	
Module No:	710	
Module Name:	IBM_TO_VAX	
Platform:	VAX	
Language:	FORTRAN	
Type:	Subroutine	
Filename:	IBM_TO_VAX.FOR	
Area:	DTF$CODE	
Issue:	3.0	
Written by:	Iain White	Date: 04/07/92
Approved by:	Iain White	Date: 04/07/92
Issued by:	Filton T.S.D.	
Document:	E12/NCA/0108	
Location:	All-in-one	
[NC_ADMIN]SYSSPECS		

Copyright
This document is copyright British Aerospace (Airbus) Limited, year as last amended.

Purpose
Transfer a file from IBM mainframe to VAX, recalling from migration if necessary.
This module is a subroutine called from: DTF_JOB.FOR. See document E12/NCA/0098.

Passed parameters:

JOBID	CHARACTER*15	Job ID 1-3 type of transfer 4-15 VAX name
FILE	CHARACTER*40	IBM dataset name
NAME	CHARACTER*12	VAX file name
JOBNO	CHARACTER*8	Job number including ACES i.e. ACES0021

Other files used:

DTF$DATA:DTF.IDX DTF	Job index
DTF$DATA:jobno.JCL	IBM TSO JCL file (Create and submit on queue RMT9RD1)
DTF$DATA:jobid.JOG	Log file for Remote Job Entry queue submit

Symbols used:

DTF$DORJE	Use RJE recall if set to "YES"
DTF_ID	IBM user ID and password

Notes
On 17/3/92 Iain White found that a DTF OPEN does an automatic HRECALL on a migrated IBM dataset. There for an error opening the file indicates a file not found without having to waste time with the RJE. As this is an undocumented feature a symbol switch has been added if DTF$DORJE equal "YES" the RJE is executed after an open failure.

Specification
Get the value of 'use RJE flag' symbol DTF$DORJE
If symbol not set use INDEX_UP function to set job status to
 'FAILED','A10 ERROR GET DTF$DORJE SYMBOL'
 Abort program

Get the IBM user ID and password from symbol DTF_ID
If symbol not set use INDEX_UP function to set job status to
 'FAILED','A40 GET DTF_ID SYMBOL'
 Abort program

Use INDEX_UP function to set job status to
 'CHECKING','PREPARING IBM FILE FOR COPY'

Build up IBM dataset file name
 Format "FILT::"fileIBM_ID where IBM_ID id the value of DTF_ID

Build VAX filename based on type of transfer
Transfer type is first three characters of JOBID
 Transfer types are TAP, APT or CLT
 VAX filename is DTF$DATA:name.TAP, (.DAT or .CLT)

See if dataset is on IBM using LIB$FIND_FILE

Dataset not found in IBM catalogue. Return status of RMS$_FNF
Use INDEX_UP function to set job status to
 'FAILED','NOT FOUND ON IBM'
 Abort program

Any other error from LIB$FIND_FILE dataset could be off line
IF RJEFLAG set to "YES" forget copy and create JCL recall file
 Use INDEX_UP function to set job status to
  'REQUEST','REQUEST RECALL'

 Use INDEX_UP function to set job status to
  'BUILDING','BUILDING RECALL JCL'

 Create JCL recall file DTF$DATA:jobno.JCL.
 If open failed use INDEX_UP function to set job status to
  'FAILED','A50 OPENING .JCL ' IOS
  Abort program

 Write JCL file using IOS to check each write
 If write error use INDEX_UP function to set job status to
  'FAILED','A?? WRITE ERROR ' IOS where ?? is the error number and IOS is the IOS error number
  Abort program

 Lines of JCL code to issue a TSO dataset recall:

```
//jobno JOB E043,CLASS=Y,MSGLEVEL=(1,0)
/*ROUTE  XEQ  FILT
//TSO   EXEC PGM=ADFMDF03,PARM=VALID,
//      DYNAMNBR=25,REGION=2000K,TIME=11
//SYSPROC DD DSN=BAE.CLIST,DISP=SHR
//SYSHELP DD DSN=BAE.HELP,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSTSIN DD *
 HRECALL "'file"
//JOBREG DD DSN=&&JCL,DISP=(,PASS),UNIT=DISK,
//       SPACE=(TRK,(1,1)),
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
/*
```

 Close JCL file

 Use INDEX_UP function to set job status to
  'RJE','SUBMITTING JCL'
 Spawn submit JCL file to Remote Job Entry queue
  'SUBMIT/SNA/QUE=RMT9RD1/LOG=DTF$DATA:jobid.JOG DTF$DATA:jobno.JCL
 Check spawn status

If not SS$_NORMAL use INDEX_UP function to set job status to
'FAILED','A180 RJE SPAWN ERROR'
Abort program
End program

Not using RJE recall (relining on auto recall)
Dataset should be on line but is it locked, Try to open it read-only and see
Close and check IOS status
If can't open use INDEX_UP function to set job status to
'STARTED','RESUBMIT (LOCK)'
Abort program

Use INDEX_UP function to set job status to
'COPYING','TRANSFERRING FILE VIA DTF'
Copy file to VAX area using COPY_FILE function
Check status of copy
If copy failed use INDEX_UP function to set job status to
'STARTED','RESUBMITTING'
Abort program
else if copy OK use INDEX_UP function to set job status to
'ARRIVED','FILE HAS ARRIVED ON VAX'
End program

2-TEST PROCEDURE

Project:	ACES DTF File Transfer
Module No:	710
Module Name:	IBM_TO_VAX
Issue:	3.1
Written by:	Iain White Date: 09/07//92
Approved by:	Iain White Date: 10/07/92
Issued by:	Filton T.S.D.
Document:	E12/NCA/0112
Location:	All-in-one [NC_ADMIN]TESTS

Run CMS_CHECK on IBM_TO_VAX.FOR

Check module is part of Project Definition File : DTF.PDF

Visually check listing IBM_TO_VAX.FOR

Vitrify detach process DTF_JOB is running

Remove symbol DTF$DORJE and submit a job
- Verify abort
- Verify DTF index holds "FAILED" and error "A10 ERROR GET DTF$DORJE SYMBOL"
- Restore DTF$DORJE symbol

Remove DTF_ID symbol Run and submit a job
- Verify abort
- Verify DTF index holds "FAILED" and error "A40 ERROR GET DTF_ID SYMBOL"
- Restore DTF_ID symbol

Create test files on IBM
- NT.A09TEST.KONFR3.DEV
- NT.A10TEST.KONFR3.PROD
- FCAE.FAPT.A11TEST.DATA
- NT.A12TEST.KONFR3.DEV
- NT.A13TEST.KONFR3.DEV and migrate
- NT.A14TEST.KONFR3.DEV and migrate
- NT.A15TEST.KONFR3.DEV and migrate

Test working by submit job tape image

Tape image name	A09TEST
Post Processor Name	KONFR3
Tape image status	DEV
IBM tape dataset	NT.A09TEST.KONFR3.DEV

- Verify DTF index set to "CHECKING" "PREPARING IBM FILE FOR COPY"
- Verify DTF index set to "COPYING" "TRANSFERRING FILE VIA DTF"
- Verify DTF index set to "ARRIVED" "FILE HAS ARRIVED ON VAX"
- Check file arrives in DTF$DATA:A09TEST.TAP

Tape image name	A10TEST
Post Processor Name	KONFR3
Tape image status	PROD
IBM tape dataset	NT.A10TEST.KONFR3.PROD

- Verify DTF index set to "CHECKING" "PREPARING IBM FILE FOR COPY"
- Verify DTF index set to "COPYING" "TRANSFERRING FILE VIA DTF"
- Verify DTF index set to "ARRIVED" "FILE HAS ARRIVED ON VAX"
- Check file arrives in DTF$DATA:A10TEST.TAP

APT source name A11TEST
IBM tape dataset FCAE.FAPT.A11TEST.DATA
Verify DTF index set to "CHECKING" "PREPARING IBM FILE FOR COPY"
Verify DTF index set to "COPYING" "TRANSFERRING FILE VIA DTF"
Verify DTF index set to "ARRIVED" "FILE HAS ARRIVED ON VAX"
Check file arrives in DTF$DATA:A11TEST.DAT

Check can not copy locked (renamed) IBM dataset
Tape image name A10TEST
Post Processor Name KONFR3
Tape image status PROD
IBM tape dataset NT.A10TEST.KONFR3.PROD
Verify program abort
Verify DTF index set to "FAILED" "NOT FOUND ON IBM"

Check can not copy locked (opened) IBM dataset
Tape image name A12TEST
Post Processor Name KONFR3
Tape image status DEV
IBM tape dataset NT.A12TEST.KONFR3.DEV
Verify program abort
Verify DTF index set to "STARTED" "RESUBMIT (LOCK)"

Check IBM off-line retrieval
Tape image name A13TEST
Post Processor Name KONFR3
Tape image status DEV
IBM tape dataset NT.A13TEST.KONFR3.DEV
Verify DTF index set to "CHECKING" "PREPARING IBM FILE FOR COPY"
Verify DTF index set to "REQUEST" "REQUEST RECALL"
Verify DTF index set to "COPYING" "TRANSFERRING FILE VIA DTF"
Verify DTF index set to "ARRIVED" "FILE HAS ARRIVED ON VAX"
Check file arrives in DTF$DATA:A13TEST.TAP

Test RJE recall. Set DTF$DORJE to "YES"
Tape image name A14TEST
Post Processor Name KONFR3
Tape image status DEV
IBM tape dataset NT.A14TEST.KONFR3.DEV
Verify DTF index set to "BUILDING" "BUILDING RECALL JCL"
Verify JCL file created DTF$DATA:jobno.JCL
Verify contents of JCL file
Verify job is submitted on queue RMT9RD1
Verify DTF index set to "REQUEST" "REQUEST RECALL"

Test for RJE queue errors. Close queue RMT9RD1
Tape image name A15TEST
Post Processor Name KONFR3

Tape image status DEV
IBM tape dataset NT.A15TEST.KONFR3.DEV
Verify DTF index set to "BUILDING" "BUILDING RECALL JCL"
Verify JCL file created DTF$DATA:jobno.JCL
Verify contents of JCL file
Verify program abort
Verify DTF index set to "FAILED" "A180 RJE SPAWN ERROR"
Open queue RMT9RD1

Proof read user manual approved	by	
Proof read system manual approved	by	
System testing	--/--/--	--/--/--
Install in test area using Project Manager	--/--/--	
Quality Assurance testing	--/--/--	--/--/--
User testing	--/--/--	--/--/--
Installed in production	--/--/--	
Issue user manual	--/--/--	

3-USER MANUAL

Title:	User guide to using ACES DTF File transfer
Issue:	2.2
Written by:	Iain White Date: 07/08/92
Approved by:	Iain White Date: 07/08/92
Issued by:	Filton T.S.D.
Document:	E12/NCA/0116
Location:	All-in-one [NC_ADMIN]USERGUIDES

Copyright
This document is copyright British Aerospace (Airbus) Limited, year as last amended.

Disclaimer
Every effort has been taken to ensure that information contained in this document is correct at the time of issue. However since new releases of this document may be periodically released, persons acting on information in this document should ensure that it has not been superseded before proceeding on the basis of statements contained within. Details of the latest issue of the document may be obtained from your key user or ACES Support, Technical Systems Development, Filton.

Preface
This document details the use of IBM to VAX file transfers under ACES.

ACES DTF FILE TRANSFER FROM IBM TO VAX

Warning

Due to this being a *FULLY* automatic process it is of the utmost importance that any file on the IBM that is involved in any stage of the transfer, must not be accessed by any means on the IBM. Also please note that due to the intense processing involved transfers can take a long time. This module is meant to be run well in advance of the images being required. It is not interactive and is subject to loading, that is the more jobs entered at any one time the longer the cumulative processing time. This is somewhat offset by the multitasking nature of the module and a deliberate bias for the module to finish jobs near the end of their cycle.

Purpose

Historically, N.C. work at Filton was carried out on IBM. ACES is now available on VAX and in order to utilise IBM work already existing within ACES, the ACES DTF transfer routines were written.

Three types of dataset on the Filton IBM have been identified as required by applications on the VAX, the three types being:-

- Tape image files.
- APT source code (including Geometry Segment files).
- Cutter Location Files (CLFILEs).

The routines use the SNA/DTF gateway between the VAX and the IBM to pull the data from the IBM onto the VAX. All additional processing is done on the VAX.

In order to prevent preceding versions of the data being used, any files transferred are flagged on the IBM to indicate that the VAX now holds the master copy. The IBM data will therefore be inaccessible for normal use. It is assumed that no further use will be made of this data and that the files are retained only as a backup.

Overview

The transfer can be seen as six separate tasks. Each of these tasks is recorded in the DTF index and is actioned by a continually running detached process.

Task 1	Is to put the transfer request on the index.
Task 2	Is to check if the appropriate dataset is currently on-line on the IBM, if not, to recall it.
Task 3	Copy the IBM dataset to a temporary VAX file.
Task 4	Reformat the IBM dataset into VAX format.
Task 5	If an APT source, make sure all GMSEGs are also copied onto the VAX.
Task 6	Move the temporary file to its destination location on the VAX.

N.B. Due to the nature of the DTF index, jobs are actioned in an alphabetical order. But as several of these tasks are carried out in batch, many jobs can be processed consecutively.

ALTERNATIVE DESTINATIONS

For each type of transfer, an optional move facility has been made available. This option is not configurable by the end user, but is set up by the ACES System Manager as an alternative menu option. The destination may be a node name or a logical of twenty characters or less. This location is displayed where the Worktype would normally appear. This is a copy to the remote node and does not lock the IBM dataset.

As a support function to VERICUT there is also a copy to SUN option, this does not lock the original IBM Dataset, and leaves the tape in IBM format, but without the checksums in column 81-84 inclusive.

FILE NAMING CONVENTIONS

Tape Images
A Tape image file on the IBM is of the format:

NT.Filename.postprocessor.status

i.e. NT.A0111020.C1A0.KONFR3.DEV

On the VAX, the tape is stored as a PPAS TESTPART under the specified WORKTYPE.

i.e. Testpart name - A01110201A0.TAP

On the SUN, the tape is stored in the specified directory. The file name is of the format:

Filename_Postprocessor.IBM

i.e. A01110201A0_KONFR3.IBM

Note that the IBM file name and name continuation fields are concatenated in the VAX/SUN file name.

APT Source
An APT source file on the IBM is of the format:

FCAE.FAPT.filename.DATA

i.e. FCAE.FAPT.A0423501.DATA

On the VAX, the program is placed on the APT index and the PDS file is created in the part program library directory.

APT$DISK:[APT140]filename.DAT &
APT$DISK:[APT140.NCTEST]filename.PDS

i.e. APT$DISK:[APT140]A0423501.DAT

Note that the .DAT file is deleted after the source is placed on the APT index.

GENERAL LIMITATIONS

The job index will never be cleared, all jobs remaining, whether they have completed successfully or failed.

A job can only exist on the index once. Jobs at completed or failed status may be re-submitted, these re-submissions overwrite any existing index entry.

APT source files cannot be pulled more than once, this is due to software checks against the APT index.

The software tries to keep track of the current job, type and VAX file name and displays this were appropriate. This is only valid for the current session.

The time fields on the forms are updated every minute and hence may be upto two minutes out. This field in not updated regularly if the VAX is under extreme load.

The datasets on the IBM are presumed to be genuine IBM datasets to the format used under IAPPS. There is no allowance for 'foreign' files stored on the IBM.

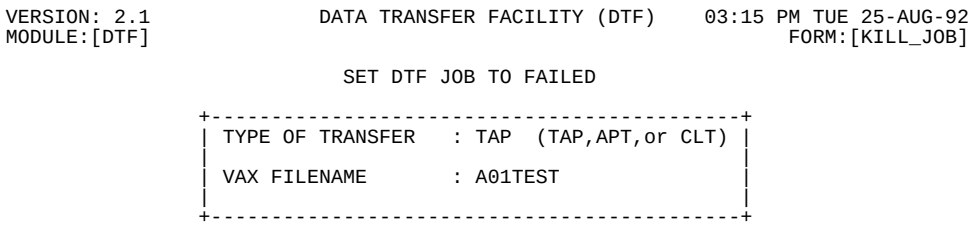
Transfers to the SUN are via SUN DNI and can only work if the SUN is switched on.

The transfer program uses the IBM userid and password of NCPPGT to access the IBM.

> Note that it is of the utmost importance that this userid and password is maintained for use by SNA/DTF.
> If the password is altered, Computer Services must be informed of the new password, so that the internal symbol references can be altered.

FAIL DTF JOB

This option is provided as a fail-safe override. Select the menu option 'FJ Fail Job'. If an unpredictable error affects the job in between statues, it may jam at a status that will never finish. This utility will reset any job to a status of failed allowing it to be re-submitted. The standard ACES FMS keys apply on this screen. The type of transfer field has a pop-up menu. If available the current job is filled in on entering the form.


VERSION: 2.1 DATA TRANSFER FACILITY (DTF) 03:15 PM TUE 25-AUG-92
MODULE:[DTF] FORM:[KILL_JOB]
SET DTF JOB TO FAILED
TYPE OF TRANSFER : TAP (TAP,APT,or CLT)
VAX FILENAME : A01TEST

MONITORING A DTF JOB

Select the menu option 'MJ Monitor DTF job'. The only way to exit from this option is to press <CTRL Z>.

A FMS screen is displayed that prompts for the type of transfer, (TAP, APT or CLT referring to tape image file, APT source file and CLFILE respectively). The program then requests the VAX filename part of the job ident, i.e. A01TEST. The status of the DTF queue is checked and displayed on the third line of the screen. The queue status consists of a list of key words separated by spaces, each of which is detailed in Appendix A. In the unlikely event of the queue status being greater than eighty characters long, it will not fit on the screen and the message:

'Queue status too complicated to be displayed'

will be displayed in its place.

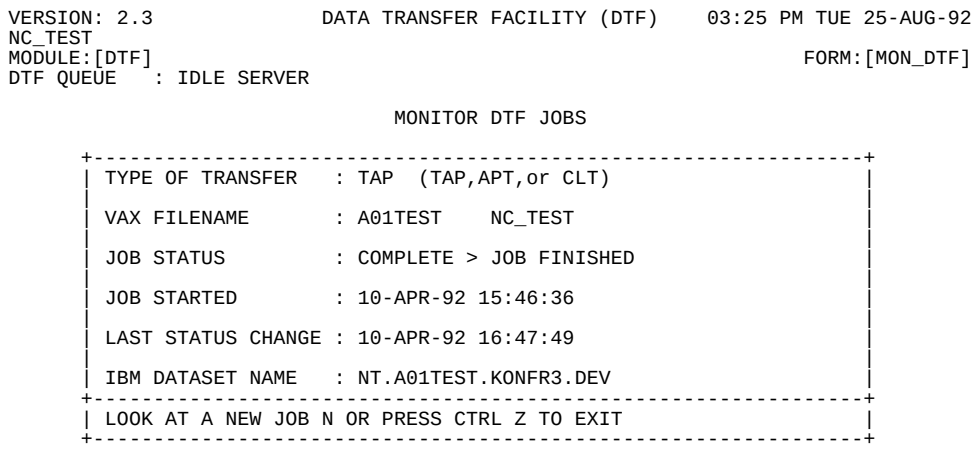
The standard ACES FMS keys apply to this screen, and when this information has been entered <RETURN> should be pressed to display the selected jobs current state. The Type of transfer field as a pop up menu.

The program interrogates the index and displays the current status of the requested job. This sampling is a continuous, real time picture of the index, and is updated once every second. To indicate that the monitor is in sampling mode, a count down timer appears on the second line in the top right hand corner of the screen. Typing <Y> at any time whilst in the sampling mode exits the sample and requests a new transfer type and job name. The DTF queue status is also checked and updated.

As this sampling is C.P.U. intensive the program times out after ten minutes in sample mode. Note the count down time indicates the number of seconds left to time out. If the VAX is under extreme load the counter my jump down.

If the job to be monitored is an APT source file the screen will change to 132 columns and will display the names and statuses of any outstanding GMSEG (include) files in an additional window.

See screen on next page


VERSION: 2.3 DATA TRANSFER FACILITY (DTF) 03:25 PM TUE 25-AUG-92
NC_TEST
MODULE:[DTF] FORM:[MON_DTF]
DTF QUEUE : IDLE SERVER
MONITOR DTF JOBS
TYPE OF TRANSFER : TAP (TAP,APT,or CLT)
VAX FILENAME : A01TEST NC_TEST
JOB STATUS : COMPLETE > JOB FINISHED
JOB STARTED : 10-APR-92 15:46:36
LAST STATUS CHANGE : 10-APR-92 16:47:49
IBM DATASET NAME : NT.A01TEST.KONFR3.DEV
LOOK AT A NEW JOB N OR PRESS CTRL Z TO EXIT

RETURNED INFORMATION

The following information is returned by the monitor:-

The JOB STATUS field contains a brief description denoting the status of a job.

STARTED	-	Job submitted or re-submitted
CHECKING	-	Interrogating IBM
REQUEST	-	Request a recall on IBM
RETRIEVE	-	Recalling on IBM
COPYING	-	Doing SNA/DTF copy
ARRIVED	-	File now on VAX
REFORMAT	-	Reformatting file
ON HOLD	-	APT source waiting for GMSEG processing
HOLDING	-	APT source waiting for one or more GMSEGs
MOVING	-	Moving temporary file to proper place on VAX or Moving temporary file to proper place on SUN
LOCK IBM	-	Locking IBM dataset
COMPLETE	-	Job has successfully finished
FAILED	-	An error has occurred (See Failed notes)

The monitor also clarifies the status with an appropriate message string. The start date and time of the job and the date and time of the last status change are also shown. The IBM dataset name is also displayed.

If the job is a tape, the destination Worktype is displayed next to the tape name. If the job is being copied to a remote area, the remote areas logical appears next to the VAX file name. If the job is being copied to the SUN, an appropriate message to this end is displayed on the form.

FAILED ERROR MASSAGES

When a job fails, the error message will show what went wrong. If the error occurred within a routine the error will have the following format:

i.e. C12 OPENING DTF INDEX 32

Where:

C	is an optional module id.
12	is the error number
OPENING DTF INDEX	is the error description
32	is an optional return status

SHOW DTF QUEUE

Select the menu option 'SQ Show DTF Queue'. The only way to exit from this option is to press <CTRL Z>. The screen is cleared and a header showing the queue name and the current date and time is displayed.

Under this option, the output similar to a standard DCL 'SHOW QUEUE' of the DTF queue. This output is continuously updated until <CTRL Z> is pressed or the program has been running for over ten minutes. This module is only included to allow the user to see where a job is in relation to the rest of the work load on the queue.

As this monitoring is C.P.U. intensive the program times out after ten minutes in sample mode. Note the count down time indicates the number of seconds left to time out. If the VAX is under extreme load the counter my jump down.

TRANSFERRING A TAPE IMAGE FILE

Select the menu option 'CT Copy Tapelib file'. A FMS screen is displayed which prompts the user for the VAX filename of the tape image file required (up to 12 alphanumeric characters) followed by the name of the post processor used (up to 6 alphanumeric characters), finally the user must specify the tape image's status as either 'PROD' or 'DEV' for production (punched) and development (un-punched) respectively. There is also an optional Worktype field, which defaults to the current Worktype.

The standard ACES FMS keys apply. As the information is entered, the IBM dataset name is built up and displayed on the screen, giving a visual check of the file to be copied. After entering the above information, the user must press <RETURN> to add the job onto the DTF index. The Worktype field and the status field have pop up menus

Before prompting for the filename, the program checks to see if the DTF queue is running. If the queue is not working, the queue status will change from 'OK' to 'STOPPED'. This does not prevent jobs from being added to the index but is merely an indication of the queue's current state.

Notes

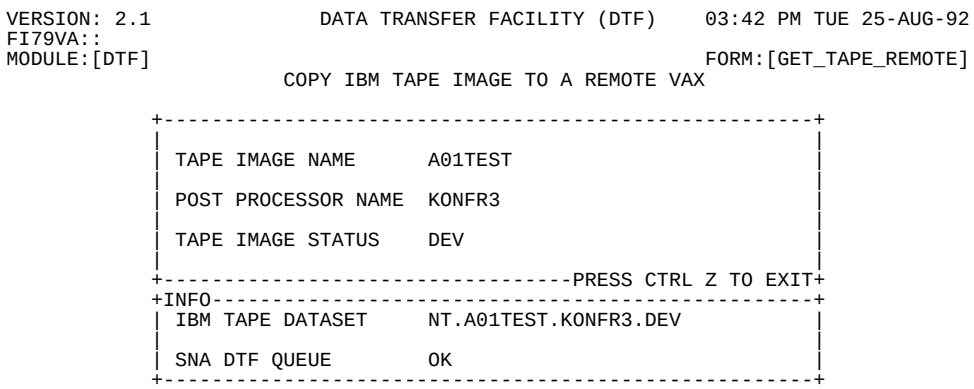
The Worktype field is not available on remote area copies nor is it available on SUN copies. For normal use the current Worktype is assumed as the Worktype to store the tape under in PPAS, but this can be overridden by the first field. The user must have the privilege to change to the specified Worktype. Both the remote copy and the SUN copy can also request tapes at a status of 'PPAS'. These are Hatfield locked tape images. For the SUN copy the screen is slightly different and an indication to the availability of the SUN node is given.

```
VERSION: 2.1              DATA TRANSFER FACILITY (DTF)    03:28 PM TUE 25-AUG-92
NC_TEST
MODULE:[DTF]                                                      FORM:[GET_TAPE]
                          COPY IBM TAPE IMAGE TO VAX

            +-------------------------------------------------------+
            |                                                       |
            | TAPE IMAGE NAME      A01TEST                          |
            |                                                       |
            | POST PROCESSOR NAME  KONFR3                           |
            |                                                       |
            | TAPE IMAGE STATUS    DEV                              |
            |                                                       |
            +---------------------------------PRESS CTRL Z TO EXIT+
            +INFO-----------------------------------------------+
            | IBM TAPE DATASET     NT.A01TEST.KONFR3.DEV            |
            |                                                       |
            | SNA DTF QUEUE        OK                               |
            +-------------------------------------------------------+
```

```
VERSION: 2.1             DATA TRANSFER FACILITY (DTF)    03:40 PM TUE 25-AUG-92
MODULE:[DTF]                                                FORM:[GET_TAPE_SUN]
                          COPY IBM TAPE IMAGE TO SUN

            +-----------------------------------------------------+
            |                                                     |
            | TAPE IMAGE NAME      A01TEST                        |
            |                                                     |
            | POST PROCESSOR NAME  KONFR3                         |
            |                                                     |
            | TAPE IMAGE STATUS    DEV                            |
            |                                                     |
            +----------------------------------PRESS CTRL Z TO EXIT+
            +INFO-------------------------------------------------+
            | IBM TAPE DATASET     NT.A01TEST.KONFR3.DEV          |
            |                                                     |
            | SUN ON LINE          YES     FI06UA::HOME:[VCT]     |
            |                                                     |
            | SNA DTF QUEUE        OK                             |
            +-----------------------------------------------------+
```



```
VERSION: 2.1             DATA TRANSFER FACILITY (DTF)    03:42 PM TUE 25-AUG-92
FI79VA::
MODULE:[DTF]                                                FORM:[GET_TAPE_REMOTE]
                    COPY IBM TAPE IMAGE TO A REMOTE VAX

            +-----------------------------------------------------+
            |                                                     |
            | TAPE IMAGE NAME      A01TEST                        |
            |                                                     |
            | POST PROCESSOR NAME  KONFR3                         |
            |                                                     |
            | TAPE IMAGE STATUS    DEV                            |
            |                                                     |
            +----------------------------------PRESS CTRL Z TO EXIT+
            +INFO-------------------------------------------------+
            | IBM TAPE DATASET     NT.A01TEST.KONFR3.DEV          |
            |                                                     |
            | SNA DTF QUEUE        OK                             |
            +-----------------------------------------------------+
```

TRANSFERRING AN APT SOURCE FILE

Select the menu option 'CI Copy IAPPS source file'. A FMS screen is displayed which prompts the user for the VAX filename of the APT file required (up to 8 alphanumeric characters). The IBM APT dataset name is built up and displayed on the screen, giving a visual check of the file to be copied.

Having entered the above information, the user must press <RETURN> to add the job to the DTF index. Standard ACES FMS keys may be used.

Before prompting for the filename, the program checks to see if the DTF queue is running. If the queue is not working, the queue status changes from 'OK' to 'STOPPED'. This does not stop jobs from being added to the index but is merely in indication of the queues current state.

The file name entered must not already be on the APT index, the program checks for this condition and returns a warning message.

```
VERSION: 2.1             DATA TRANSFER FACILITY (DTF)    03:45 PM TUE 25-AUG-92
NC_TEST
MODULE:[DTF]                                                      FORM:[GET_APT]

                          COPY IBM APT SOURCE TO VAX

            +-------------------------------------------------------+
            |                                                       |
            | APT SOURCE NAME      A01TEST                          |
            |                                                       |
            +---------------------------------PRESS CTRL Z TO EXIT+

            +INFO-----------------------------------------------+
            | IBM TAPE DATASET     FCAE.FAPT.A01TEST.DATA       |
            |                                                   |
            | SNA DTF QUEUE        OK                           |
            +---------------------------------------------------+
```

APPENDIX A: QUEUE STATUS KEYWORDS

ALIGNING:	Queue is printing alignment pages.
CLOSED:	Queue is closed and will not accept new jobs until the queue is put in an open state.
IDLE:	Queue contains no job requests.
LOWERCASE:	Queue is associated with a printer that can print both upper and lower-case characters.
PAUSED:	Execution of all current jobs in queue is temporarily halted.
PAUSING:	Queue is temporarily halting execution. Currently executing jobs are completing; no new jobs can begin executing.
REMOTE:	Queue is assigned to a physical device that is not connected to the local node.
RESETTING:	Queue is resetting and stopping.
RESUMING:	Queue is restarting after pausing.
SERVER:	Queue processing is directed to a server symbiont.
STALLED:	Physical device to which queue is assigned is stalled; that is, the device has not completed the last I/O request submitted to it. Most likely for a printer is 'out of paper'.
STARTING:	Queue is starting.
STOPPED:	Queue is stopped.
STOPPING:	Queue is stopping.
UNAVAILABLE:	Physical device to which queue is assigned is not available.
UNKNOWN:	Could not find queue status.

APPENDIX B: FORMAT CONVERSIONS

APT Source Dataset
All occurrences of ££GMSEG are replaced with %INCLUDE.
A MACHIN/VAX statement is added as line two if line one is a PARTNO.
A source file may only contain 99 GMSEG statements.

Tape Image
The IBM tape image is converted in accordance with the Chadderton document 'BAe(CA) NC Tape Image VAX Storage Format Definition' Ref: MS/AS/DEFN/0110 Issue 2 if the destination is on the VAX. The IBM tape image is assumed to be of the format outlined in 'NC Tape Image Storage Format On Filton IBM' Ref: EP/48. The SUN tape images are as the IBM format but with the checksums removed.

Audit Trail
All conversion routines record their name and version along with the date and time of conversion in the converted file.

APPENDIX C: ACES FMS KEYS

PF1	Known as the 'GOLD' key, used in conjunction with other keys.
PF2 OR HELP	Provides help. First press displays a one line message, further presses will display full screen(s) if available.
PF3	Characters entered in field will overwrite existing characters.
GOLD PF3	Characters entered in field will be inserted.
RETURN	Input/selection is complete, (form filled in).
CTRL Z	Exit from input/selection area (quit).
CTRL P	Print the current screen to the default printer.
CTRL A	List modules making up current program image.
GOLD D	Put default value in current field.
LF	(Line Feed) Blank current field.
GOLD L	Put last entered value into current field.
GOLD B	Blank all fields in input area, inserting any active defaults.
GOLD M	Pop-up menu. Some fields may have menus that allow entry to be chosen using MENU_DRIVER input and placed in the corresponding fields.
TAB OR DOWN ARROW	Moves to the next input field on the form. If the field is the last one, the cursor jumps to the first field automatically.
BS OR UP ARROW	Moves to the previous input field on the form. If the field is the first one, the cursor jumps to the last field automatically.
GOLD LEFT ARROW	Move to field furthest to the left of current field.
GOLD RIGHT ARROW	Move to field furthest to the right of current field.
GOLD UP ARROW OR PREV SCREEN	Go to first field within the input area.
GOLD DOWN ARROW OR NEXT SCREEN	Go to last field within the input area.
GOLD S	Moves cursor to the start of the current field.
GOLD E	Moves cursor to the position after the last character in the current field.

4-SYSTEM MANUAL

Project:	ACES DTF File Transfer
Module No:	710
Module Name:	IBM_TO_VAX
Platform:	VAX
Language:	FORTRAN
Type:	Subroutine
Filename:	IBM_TO_VAX.FOR
Area:	DTF$CODE
Issue:	7.0
Written by:	Iain White Date: 04/08/92
Approved by:	Iain White Date: 06/08/92
Issued by:	Filton T.S.D.
Document:	E12/NCA/0114
Location:	All-in-one [NC_ADMIN]SYSGUIDE

Copyright
This document is copyright British Aerospace (Airbus) Limited, year as last amended.

Disclaimer
Every effort has been taken to ensure that information contained in this document is correct at the time of issue. However since new releases of this document may be periodically released, persons acting on information in this document should ensure that it has not been superseded before proceeding on the basis of statements contained within. Details of the latest issue of the document may be obtained from your key user or ACES Support, Technical Systems Development, Filton.

Purpose
Transfer a file from IBM mainframe to VAX, recalling from migration if necessary.
This module is a subroutine called from: DTF_JOB.FOR. See document E12/NCA/0098.
There are two entry points in DTF_JOB corresponding to DTF statuses of "STARTED" and "RECALLED".

Passed parameters:

JOBID	CHARACTER*15	Job ID 1-3 type of transfer 4-15 VAX name
FILE	CHARACTER*40	IBM dataset name
NAME	CHARACTER*12	VAX file name
JOBNO	CHARACTER*8	Job number including ACES i.e. ACES0021

Other files used:

DTF$DATA:DTF.IDX DTF	Job index
DTF$DATA:jobno.JCL	IBM TSO JCL file (Create and submit on queue RMT9RD1)
DTF$DATA:jobid.JOG	Log file for Remote Job Entry queue submit

Logicals used:

FILT	IBM node name

Symbols used:

DTF$DORJE	Use RJE recall if set to "YES"
DTF_ID	IBM user ID and password

Notes
On 17/3/92 Iain White found that a DTF OPEN does an automatic HRECALL on a migrated IBM dataset. There for an error opening the file indicates a file not found without having to waste time with the RJE. As this is an undocumented feature a symbol switch has been added if DTF$DORJE equal "YES" the RJE is executed after an open failure.

Called routines:

LIB$FIND_FILE	Find file(s)	VMS Run Time Library
LIB$FIND_FILE_END	End find file(s)	VMS Run Time Library
LIB$GET_SYMBOL	Get VMS symbol	VMS Run Time Library
LIB$SPAWN	Spawn DCL command	VMS Run Time Library
INT_CHAR	Convert integer to character	Filton function
NCHARS	Length of string	General Function
INDEX_UP	Update DTF index job status record	DTF function

Include files:

GENERALSOFT:MODULES.CMN	General module common block
$SSDEF	System services definitions
$RMSDEF	Record management services definitions

File units used:

UNIT 1	Open test on IBM dataset
UNIT 1	Create JCL file

Also uses:
DEC SNA/DTF
VAX to IBM Remote Job Entry queue RMT9RD1

Types of transfer supported:

TAP	Tape image file	DTF$DATA:name.TAP
APT	APT140 file	DTF$DATA:name.DAT
CLT	Cutter Location file	DTF$DATA:name.CLT

Error codes:
All errors are reported in the DTF job index
Format of errors i.e. A50 OPENING DTF INDEX 32
Where:

A	is an optional module id.
50	is the error number
OPENING DTF INDEX	is the error description
32	is an optional return status

Job status	Error
FAILED	A10 ERROR GET DTF$DORJE SYMBOL
FAILED	A40 GET DTF_ID SYMBOL
FAILED	NOT FOUND ON IBM
FAILED	A50 OPENING .JCL IOS number
FAILED	A60 WRITE ERROR IOS number
FAILED	A70 WRITE ERROR IOS number
FAILED	A80 WRITE ERROR IOS number
FAILED	A90 WRITE ERROR IOS number
FAILED	A100 WRITE ERROR IOS number
FAILED	A110 WRITE ERROR IOS number
FAILED	A120 WRITE ERROR IOS number
FAILED	A180 RJE SPAWN ERROR

Other statuses recorded in DTF job index:

CHECKING	PREPARING IBM FILE FOR COPY'
REQUEST	REQUEST RECALL'
STARTED	RESUBMIT (LOCK)'
COPYING	TRANSFERRING FILE VIA DTF'
STARTED	RESUBMITTING'
ARRIVED	FILE HAS ARRIVED ON VAX'
BUILDING	BUILDING RECALL JCL'
RJE	SUBMITTING JCL'

Built up JCL file:

```
//jobno JOB E043,CLASS=Y,MSGLEVEL=(1,0)
/*ROUTE  XEQ  FILT
//TSO   EXEC PGM=ADFMDF03,PARM=VALID,
//      DYNAMNBR=25,REGION=2000K,TIME=11
```

```
//SYSPROC DD DSN=BAE.CLIST,DISP=SHR
//SYSHELP DD DSN=BAE.HELP,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSTSIN DD *
 HRECALL "'file"
//JOBREG DD DSN=&&JCL,DISP=(,PASS),UNIT=DISK,
//       SPACE=(TRK,(1,1)),
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
/*
```

5-VAX DCL

```
$! DEFINE_PRINTER.COM
$!------------------------------------------------------------------------------
$!
$!      ****************************************************************
$!      *                   FILTON ACES LOCAL SUPPORT.                 *
$!      *                   -------------------------                  *
$!      *                                                              *
$!      *   Copyright (c) British Aerospace (Civil Aircraft) Limited.  *
$!      *                    Date as last amendment.                   *
$!      *                                                              *
$!      *                     All rights reserved.                     *
$!      ****************************************************************
$!
$!      COMMAND FILE: DEFINE_PRINTER                                    VERSION: V2.3
$!     PURPOSE:      To determine the nearest printer to the login terminal
$!                   and define appropriate characteristics.
$!     AUTHOR:       I WHITE                                        DATE: 04/10/1989
$!     PROGRAMS:     LAT_INFO
$!     NOTES:        For interactive jobs, the printer is determined from the
$!                   translation TT: If this is of the form LTA*, LAT_INFO is used
$!                   to determine the actual server and port number. We then 'jump'
$!                   on this value to the relevant printer.
$!
$!                   For batch jobs, we use the username, and assume the users
$!                   'home' printer.
$!
$!     AMENDMENTS:
$!                   V2.0  IW 10/12/89 If application is called up by a SET HOST
$!                   mechanism, TT will be defined as an RT device name, so LAT_INFO
$!                   won't work. To overcome this, if an interactive user's terminal
$!                   cannot be determined, we will present a menu and ask.
$!                   V2.1 IW 03/01/90 Was checking for MODE.NES."BATCH" now checking
$!                   for MODE.eqs. "INTERACTIVE" , as DECNet file transfers using
$!                   username and password go through the login command file, and
$!                   have mode as "OTHER"
$!                   V2.2 IW 07/02/91 To use menu driver to select printer
$!                   V2.3 IW 21/02/91 To make general for all ACES
$!------------------------------------------------------------------------------
$!
$!      Do we have any printer definition files in the first place?
$       FILE_STATUS = F$SEARCH ("GENERALDATA:*.PRINTER_DEF")
$!
$!      No printer definition files use default printer
$       IF FILE_STATUS .EQS. "" THEN GOTO DEFAULT_PRINTER
$!
$!      Check user has been defined, if not define now
$       IF F$TYPE(PSEUDO_USER) .EQS. "" THEN RUN GENERALEXE:DEFINE_USER
$!
$!      Set printer to default
$       PRINTER :== 'DEFAULT_PRINTER'
$!
$!      Are we running in batch?
$       IF F$MODE() .EQS. "BATCH" THEN GOTO PRINTER_DEF
$!
$!      The following is a definition of some of the more common escape codes
$!      that are used in command files.
$!
$!      Firstly we define <ESC>
$       ESC           = " "
$       ESC[0,7]      = 27
$!
$       CLEAR_TO_EOL = "''ESC'[0K"                          !Clear to end o line
$       CLEAR_SCREEN = "''ESC'[2J"                          !Clear screen
$       HOME_CURSOR  = "''ESC'[0;0H"                        !Home cursor
$       CLEAR_ATTR   = "''ESC'[m"                           !Clear video attributes
$       SCROLL3_24   = "''ESC'[3;24r"                       !Set scroll region
$       SCROLL_ALL   = "''ESC'[0;24r"                       !Clear scroll region
$       BOLD_ATTR    = "''ESC'[1m"                          !Bold
$       FLASH_ATTR   = "''ESC'[5m"                          !Blink
$       UNDER_ATTR   = "''ESC'[4m"                          !Underscore
$       REVERSE_ATTR = "''ESC'[7m"                          !Reverse video
$       D_HIGHT_TOP  = "''ESC'#3"                           !Double height (top)
$       D_HIGHT_BOT  = "''ESC'#4"                           !Double height (bottom)
$       D_WIDTH      = "''ESC'#6"                           !Double width
$!
$!      Initialise variables.
$       DEVICE = "NONE"
$       PORT   = "NONE"
$!
$!      Is LAT_INFO installed? If so calculate the nearest Printer.
$       IF LAT_INFO_INSTALLED .NES. "YES" THEN GOTO PRINTER_MENU
$!
```

```
$!     First, find out what terminal this is?
$      DEVICE = F$TRNLNM("TT")
$!
$!     If it is an LTA terminal, find the server and port
$      IF (F$EXTRACT(0,3,DEVICE).EQS."RTA") THEN GOTO PRINTER_MENU
$      IF (F$EXTRACT(0,3,DEVICE).EQS."LTA") THEN $ LAT_INFO TEST_D TEST_P
$!     Force case to upper
$      DEVICE = F$EDIT(TEST_D,"UPCASE")
$      PORT   = F$EDIT(TEST_P,"UPCASE")
$!
$!     Open the printer locations file and loop around until one
$!     meets our current requirements.
$      OPEN/READ/ERR=PRINTER_MENU INFILE GENERALDATA:PRINTER_LOCATIONS.DEF
$!
$!     Read the File
$START_READ:
$      READ/ERR=READ_ERROR INFILE LINE
$!
$!     Separate out the information.
$      TEST_DEVICE  = F$EXTRACT(0,20,LINE)
$      TEST_PORT    = F$EXTRACT(20,20,LINE)
$      TEST_PRINTER = F$EXTRACT(40,20,LINE)
$!
$!     Tidy up the Data.
$      TEST_DEVICE  = F$EDIT(TEST_DEVICE, "UPCASE,COMPRESS,TRIM")
$      TEST_PORT    = F$EDIT(TEST_PORT, "UPCASE,COMPRESS,TRIM")
$      TEST_PRINTER = F$EDIT(TEST_PRINTER,"UPCASE,COMPRESS,TRIM")
$!
$      IF (TEST_DEVICE .EQS. "ANY" .OR. TEST_DEVICE .EQS. DEVICE) .AND. -
          (TEST_PORT   .EQS. "ANY" .OR. TEST_PORT .EQS. PORT) THEN -
                                                      GOTO FOUND_PRINTER
$!
$      GOTO START_READ
$!
$!     Error in read.
$READ_ERROR:
$      CLOSE INFILE
$      GOTO PRINTER_MENU
$!
$!     Found a printer
$FOUND_PRINTER:
$      CLOSE INFILE
$      PRINTER :== 'TEST_PRINTER'
$!     Look for printer definition file
$      IF F$SEARCH("GENERALDATA:''PRINTER'.PRINTER_DEF") .NES. "" THEN -
                                                      GOTO PRINTER_DEF
$!
$!     Select printer menu
$PRINTER_MENU:
$      IF PRINTER_MENU .NES. "YES" THEN GOTO PRINTER_DEF
$!
$DISPLAY_MENU:
$!     Display a menu of all the known printers
$      RUN GENERALEXE:SELECT_PRINTER
$      GOTO PRINTER_DEF
$!
$!     What is my current device?
$QUERY:
$      WRITE SYS$OUTPUT "''ESC'[23;0HDevice """,DEVICE,"""   Port """,PORT,""""
$      GOTO ASK_TEMP
$!
$!     Now for the printer definitions themselves
$PRINTER_DEF:
$      FILE_STATUS = F$SEARCH("GENERALDATA:''PRINTER'.PRINTER_DEF")
$      IF FILE_STATUS .NES. "" THEN GOTO DEFINE_IT
$!
$!     Definition file for chosen printer does not exist!
$!
$!     BATCH JOB - Write out message and take default values
$      IF F$MODE().NES."INTERACTIVE" THEN GOTO DEFAULT_PRINTER
$!
$!     Regardless of whether the system parameters say that we are not
$!     to go to the menu, if we get here and the definition file does not
$!     exist, then IF we are interactive then we must go and ask!
$      GOTO DISPLAY_MENU
$!
$!     Set up printer
$DEFINE_IT:
$      RUN_DEFINE := $GENERALEXE:DEFINE_PRINTER
$      RUN_DEFINE 'PRINTER'
$!
$END:
$      EXIT
$!
$!     Default printer
```

```
$DEFAULT_PRINTER:
$!
$!     The ultimate deterrent - default printer!
$      WRITE SYS$OUTPUT " "
$      WRITE SYS$OUTPUT " "
$      WRITE SYS$OUTPUT "***************************************"
$      WRITE SYS$OUTPUT "***************************************"
$      WRITE SYS$OUTPUT " CANNOT FIND PRINTER DEFINITION FILE!"
$      WRITE SYS$OUTPUT "  SENDING OUTPUT TO SYSTEM PRINTER"
$      WRITE SYS$OUTPUT "***************************************"
$      WRITE SYS$OUTPUT "***************************************"
$      WRITE SYS$OUTPUT " "
$      WRITE SYS$OUTPUT " "
$      PR_COLS :== 132
$      PR_LINES :== 60
$      PR_BOX_GR :== "NO"
$      GENERALPRINT :== "PRINT/NOIDENTIFY/NONOTIFY"
$      PRINTER :== "SYSTEM"
$      EXIT
```

6-VAX FORTRAN

```
C ************************************************************************
C * FILTON ACES LOCAL SUPPORT                                            *
C * Copyright (c) British Aerospace (Civil Aircraft) Limited         .....       *
C * Date as last amendment                                           *
C ************************************************************************
C ************************************************************************
C * IBM_TO_VAX                                          Version 02.03                *
C * Purpose : Transfers a file from IBM to VAX                                       *
C * Author  : Iain White                                        Date  : 30/06/92     *
C * Parameters :       JOBID - Job type/name                                         *
C *                    FILE  - IBM dataset name                                      *
C *                    NAME  - VAX file name                                         *
C *                    JOBNO - Job number including ACES i.e. ACES0021               *
C * Calls      :       INT_CHAR                                                      *
C *                    NCHARS                                                        *
C *                    INDEX_UP                                                      *
C *                    LIB$FIND_FILE                                                 *
C *                    LIB$FIND_FILE_END                                             *
C *                    LIB$GET_SYMBOL                                                *
C *                    LIB$SPAWN                                                     *
C * Also uses  :  DTF$DATA:DTF.IDX                                                   *
C *               DTF$DATA:jobno.JCL (Creates and submits ON RMT9RD1)          *
C *               DTF$DATA:jobid.JOG (Log file for JCL submit)                 *
C * Notes      :  17/3/92 IW found that a DTF OPEN does an automatic HRECALL on*
C *               a migrated IBM data set. There for an error opening the file *
C *               indicates a file not found without having to waste time with *
C *               the RJE. As this is an undocumented feature a symbol switch  *
C *               has been added if DTF$DORJE equal YES the RJE is executed    *
C *               after an open failure.                                       *
C ************************************************************************
C
       SUBROUTINE     IBM_TO_VAX(JOBID, FILE, NAME, JOBNO)

C      Declare all variables
       IMPLICIT       NONE

       CHARACTER*3           INT_CHAR                      !Filton function
       CHARACTER*3           RJEFLAG/'NO '/        !Do RJE flag
       CHARACTER*8           JOBNO                         !Job number
       CHARACTER*12          NAME                          !VAX Filename
       CHARACTER*15          JOBID                         !Passed Job ID
       CHARACTER*40          FILE                          !IBM Filename
       CHARACTER*80          IBM_ID                        !IBM id/password
       CHARACTER*255         DUMMY                         !FIND_FILE return
       CHARACTER*255         IBM_FILE                      !Full IBM file name
       CHARACTER*256         VAX_FILE                      !Full VAX file name
       INTEGER*4             NCHARS                        !General function
       INTEGER*4             STATUS                        !Return status
       INTEGER*4             IOS                           !File IO status
       INTEGER*4             CONTEXT               !Find file context
       INTEGER*4             LIB$FIND_FILE_END             !RTL
       INTEGER*4             LIB$FIND_FILE                 !RTL
       INTEGER*4             LIB$GET_SYMBOL        !RTL
       INTEGER*4             LIB$SPAWN                     !RTL
       LOGICAL        DFLAG/.FALSE./        !File delete flag
       LOGICAL        INDEX_UP                      !DTF function

       INCLUDE        'GENERALSOFT:MODULES.CMN'
       INCLUDE        '($SSDEF)/NOLIST'
       INCLUDE        '($RMSDEF)/NOLIST'


C      Set module/name
       PARAMETER (MODNO=710)
       DATA MODULES(MODNO) /'IBM_TO_VAX FV2.3'/

C      Standard format
10     FORMAT(A)

C      Get RJE symbol
       STATUS = LIB$GET_SYMBOL('DTF$DORJE',RJEFLAG)
       IF (STATUS .NE. SS$_NORMAL) THEN
         STATUS=INDEX_UP(JOBID,'FAILED',
     *''A10 ERROR GET DTF$DORJE SYMBOL')
         GOTO 999
       END IF

C      Get IBM user ID and password from symbol
       STATUS = LIB$GET_SYMBOL('DTF_ID',IBM_ID)
       IF (STATUS .NE. SS$_NORMAL) THEN
         STATUS=INDEX_UP(JOBID,'FAILED','A40 GET DTF_ID SYMBOL')
         GOTO 999
```

```
        END IF

C       Update index
        STATUS = INDEX_UP(JOBID,'CHECKING','PREPARING IBM FILE FOR COPY')

C       Build IBM filename
        IBM_FILE = 'FILT::"'//FILE(1:NCHARS(FILE))//
     *IBM_ID(1:NCHARS(IBM_ID))//'"'

C       Build VAX filename based on type of transfer
        IF (JOBID(1:3) .EQ. 'TAP')
     *VAX_FILE = 'DTF$DATA:'//NAME(1:NCHARS(NAME))//'.TAP'
        IF (JOBID(1:3) .EQ. 'APT')
     *VAX_FILE = 'DTF$DATA:'//NAME(1:NCHARS(NAME))//'.DAT'
        IF (JOBID(1:3) .EQ. 'CLT')
     *VAX_FILE = 'DTF$DATA:'//NAME(1:NCHARS(NAME))//'.CLT'

C       See if file on IBM and on line
        CONTEXT = 0
C       Reset search context
        STATUS = LIB$FIND_FILE_END(CONTEXT)
C       Search for file
        STATUS = LIB$FIND_FILE(IBM_FILE(1:NCHARS(IBM_FILE)),DUMMY
     *,CONTEXT)

C       Not found in IBM catalogue
        IF (STATUS .EQ. RMS$_FNF) THEN
C         Update index
          STATUS = INDEX_UP(JOBID,'FAILED','NOT FOUND ON IBM')
          GOTO 999
        END IF

C       Error, could be off line
        IF (STATUS .NE. RMS$_NORMAL) THEN
C         Do RJE
          IF (RJEFLAG .EQ. 'YES') GOTO 60
C         Update index
          STATUS = INDEX_UP(JOBID,'REQUEST','REQUEST RECALL')
          GOTO 999
        END IF

C       File should be on line but is it locked, Try to open and see
        OPEN (UNIT=1,FILE=IBM_FILE(1:NCHARS(IBM_FILE))
     *,STATUS='OLD',IOSTAT=IOS,READONLY)
C       Close file
        CLOSE (UNIT=1)
        IF (IOS .NE. 0) THEN
C         Update index
          STATUS = INDEX_UP(JOBID,'STARTED','RESUBMIT (LOCK)')
          GOTO 999
        END IF

C       Copy file

C       Update index
        STATUS = INDEX_UP(JOBID,'COPYING','TRANSFERRING FILE VIA DTF')
C       Copy file to VAX area
        CALL COPY_FILE(IBM_FILE(1:NCHARS(IBM_FILE))
     *,VAX_FILE(1:NCHARS(VAX_FILE)),DFLAG,STATUS)
C       Check status of copy
        IF (STATUS .LT. 0) THEN
C         Update index
          STATUS = INDEX_UP(JOBID,'STARTED','RESUBMITTING')
        ELSE
C         Update index
          STATUS = INDEX_UP(JOBID,'ARRIVED','FILE HAS ARRIVED ON VAX')
        END IF
        GOTO 999

C Jump to do RJE
60      CONTINUE
C       File is migrated, create recall JCL

C       Update index
        STATUS = INDEX_UP(JOBID,'BUILDING','BUILDING RECALL JCL')

C       Create JCL file
        OPEN (UNIT=1,FILE='DTF$DATA:'//JOBNO//'.JCL'
     *,CARRIAGECONTROL='LIST',STATUS='NEW',IOSTAT=IOS)
        IF (IOS .NE. 0) THEN
C         Update index
          STATUS=INDEX_UP(JOBID,'FAILED','A50 OPENING .JCL '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
```

```
C       Write file
        WRITE (1,10,IOSTAT=IOS) '//'//JOBNO//
     *' JOB E043,CLASS=Y,MSGLEVEL=(1,0)'
        IF (IOS .NE. 0) THEN
C         Update index
          STATUS=INDEX_UP(JOBID,'FAILED','A60 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '/*ROUTE  XEQ  FILT'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A70 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//TSO   EXEC PGM=ADFMDF03,PARM=VALID,'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A80 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//      DYNAMNBR=25,REGION=2000K,
     *TIME=11'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A90 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//SYSPROC DD DSN=BAE.CLIST,DISP=SHR'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A100 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//SYSHELP DD DSN=BAE.HELP,DISP=SHR'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A110 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//SYSTSPRT DD SYSOUT=A'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A120 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//SYSTSIN DD *'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A130 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) ' HRECALL '''//FILE(1:NCHARS(FILE))//''''
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A140 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//JOBREG DD DSN=&&JCL,DISP=(,PASS),
     *UNIT=DISK,'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A150 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//       SPACE=(TRK,(1,1)),'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A160 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '//       DCB=(RECFM=FB,LRECL=80,
     *BLKSIZE=3120)'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A170 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
        WRITE (1,10,IOSTAT=IOS) '/*'
        IF (IOS .NE. 0) THEN
          STATUS=INDEX_UP(JOBID,'FAILED','A180 WRITE ERROR '//
     *INT_CHAR(IOS))
          GOTO 999
        END IF
```

```
C       Close file
        CLOSE (UNIT=1)

C       Submit job to IBM

C       Update index
        STATUS = INDEX_UP(JOBID,'RJE','SUBMITTING JCL')

C       Spawn submit to queue
        STATUS= LIB$SPAWN('SUBMIT/SNA/QUE=RMT9RD1/LOG=DTF$DATA:'//
     *JOBID(1:NCHARS(JOBID))//'.JOG'
     *//' DTF$DATA:'//JOBNO//'.JCL')

C       Check spawn status
        IF (STATUS .NE. SS$_NORMAL) THEN
C         Update index
          STATUS=INDEX_UP(JOBID,'FAILED','A180 RJE SPAWN ERROR')
          GOTO 999
        END IF

C         Update index
          STATUS = INDEX_UP(JOBID,'REQUEST','REQUEST RECALL')


C       Jump point
999     CONTINUE

        RETURN
        END
```

7-VAX BASIC

```
!===============================================================================
!       Name:           VT$PUT_VTEXT.BAS
!       Purpose:        Put a line of text into the virtual text display
!       Project:        CLAWS
!       Class:          Virtual Text Manager
!       Sub-class:      Function
!       Author: Iain White (Clares Equipment Ltd)
!       Arguments:      Vtext name, text string, line number
!       Returns:        -1 - OK, 0 - Error
!       Notes:          Text is stored with a terminator (VTM.TERMINATOR).
!                       First character on the line is used as a flag.
!                       Second character is reserved for the tag vale
!                       (X - tagged, space non-tagged).
!                       For the above the display size is the string length + 3.
!                       To allow for scrolling the display height is the number of
!                       lines + 1
!===============================================================================


FUNCTION BYTE VTM$PUT_VTEXT(REC$VTEXT VTEXT_NAME, STRING STR, WORD ROW)

! Variables must be declared
  OPTION TYPE = EXPLICIT

! Include virtual text definitions
  %INCLUDE "STD:CIW_VTEXT.INC"

! External functions
  EXTERNAL LONG FUNCTION SMG$CHANGE_VIRTUAL_DISPLAY(LONG,LONG                    &
                                                    LONG,LONG,LONG,LONG)     !RTL
  EXTERNAL LONG FUNCTION SMG$SET_CURSOR_ABS(LONG,LONG,LONG)                      !RTL
  EXTERNAL LONG FUNCTION SMG$PUT_LINE(LONG,LONG,LONG,LONG)                       !RTL

! Local variables
  DECLARE LONG        RET_STATUS                  !Return status
  DECLARE BYTE        CHANGE                      !Display size change needed
  DECLARE WORD        STR_LEN               !Length of passed string

! Set return value to OK
  VTM$PUT_VTEXT = -1

! Get length of passed string
  STR_LEN = LEN(STR)

! Set default to no change
  CHANGE = 0

! Check if display is wide enough
  IF STR_LEN > VTEXT_NAME::WIDTH THEN
    VTEXT_NAME::WIDTH = STR_LEN
    CHANGE = -1
  END IF

! Check if there are enough lines on the display
  IF ROW > VTEXT_NAME::HEIGHT THEN
    VTEXT_NAME::HEIGHT = ROW
    CHANG = -1
  END IF

! Make sure there is a valid SMG display id in the vtext structure
  IF VTEXT_NAME::ID < 1 THEN

!   Programmer should have used a call to VTM$CREATE_VTEXT, but lets be helpful
    RET_STATUS = SMG$CREATE_VIRTUAL_DISPLAY(VTEXT_NAME::HEIGHT + 1,          &
                                            VTEXT_NAME::WIDTH + 3, VTEXT_NAME::ID,,,)

!   Check return status (probably run out of memory!)
    IF (RET_STATUS AND 1) <> 0 THEN
      VTM$PUT_VTEXT = 0
    END IF

  ELSE

!   Do we need to change the display size?
    IF CHANGE THEN
C     Change SMG display
      RET_STATUS = SMG$CHANG_VIRTUAL_DISPLAY(VTEXT_NAME::ID,                     &
                                             VTEXT_NAME::HEIGHT + 1,             &
                                             VTEXT_NAME::WIDTH + 3,,,)

!     Check return status (probably run out of memory!)
      IF (RET_STATUS AND 1) <> 0 THEN
```

```
        VTM$PUT_VTEXT = 0                   ! Error
      END IF

    END IF

  END IF

! Are we still OK
  IF VTM$PUT_VTEXT THEN
!   Position cursor at appropriate line
    RET_STATUS = SMG$SET_ABS(VTEXT_NAME::ID,INTEGER(ROW,LONG),3)
!   Write text to SMG display, with line advance
    RET_STATUS = SMG$PUT_LINE(VTEXT::ID,STR + CHR$(VTM.TERMINATOR), 1,)
!   Update current cursor position in vtext structure
    VTEXT_NAME::ROW = ROW
    VTEXT_NAME::COLUMN = 3
  END IF

! Check return status
  IF (RET_STATUS AND 1) <> 0 THEN
    VTM$PUT_VTEXT = 0                ! Error
  END IF

END FUNCTION
```

```
*****************************************************************************
* FILTON ACES LOCAL SUPPORT
* Copyright (c) British Aerospace (Civil Aircraft) Limited
* Date as last amendment
*****************************************************************************
*****************************************************************************
* Name       : DBASE_EDIT                                                    *
* Version    : 01.01                                                         *
* Purpose    : Performs basic updates on the PANOPLIE database               *
* Parameters : None                                                          *
* Calls      : FDV$DISP                                                      *
*              FDV$PUTL                                                      *
*              FDV$PUT                                                       *
*              FDV$GET                                                       *
*              FDV$CLEAR                                                     *
* Programs   : None                                                          *
* Symbols    : None                                                          *
* Logicals   : None                                                          *
* Also uses  : Database file: PROD$DEV1:[PANOPLIE.DBASE]CNCSCHEMA.ROO        *
*              Storage area CNC01: PROD$DEV1:[PANOPLIE.DBASE]CNC01.DBS       *
* Notes      : This module is only called from PANOPLIE_SCROL_IO.FOR         *
*              called from within FMS display form.                          *
*****************************************************************************

IDENTIFICATION DIVISION.
************************
PROGRAM-ID.                 DBASE_EDIT INITIAL.
AUTHOR.                     I. WHITE.
INSTALLATION.               FI79VG BAE FILTON.
DATE-WRITTEN.               20/02/1989.

ENVIRONMENT DIVISION.
*********************

CONFIGURATION SECTION.
*----------------------

SPECIAL-NAMES.

       CONSOLE IS CONSOLE.

DATA DIVISION.
**************

SUB-SCHEMA SECTION.
*------------------

* Set database
DB DEFAULT_SUBSCHEMA WITHIN CNC_SCHEMA FOR "CNCDATABASE".

* Set keeplist
LD KEEP-104.
LD KEEP-102.
LD KEEP-VALID-LINK.

WORKING-STORAGE SECTION.
*-----------------------

          01 CMS-SWITCH              PIC 9.
             88 CMS-FOUND            VALUE 1.
           01 EOS                    PIC 9.
             88 EOS1                 VALUE 1.
           01 MAT                    PIC X(7).
           01 THICK                  PIC X(4).
           01 OLDVALUE               PIC X(23).
           01 FFT-TRM         PIC 9(7)  COMP.
           01 ENTERED         PIC 9(7)  COMP VALUE 0.
           01 TAB                    PIC 9(7)  COMP VALUE 11.
           01 WORKSPACE              PIC X(12).
           01 CHECK_WORKSPACE PIC 9(5)  COMP VALUE 2000.
           01 LOGICAL_UNIT           PIC 9(5)  COMP VALUE 3.
           01 LOGICAL_UNIT_TT PIC 9(5)  COMP VALUE 2.
           01 TERMINAL_AREA   PIC 9(5)  COMP VALUE 12.
           01 TERMINAL_SIZE   PIC X(12).

LINKAGE SECTION.
*---------------

*         Parameters passed from FORTRAN
           01 PART                   PIC X(23).
           01 TOOL                   PIC X(22).
           01 CMS                    PIC X(8).
```

```
           01 MATERIAL              PIC X(7).
           01 THICKNESS             PIC X(4).
           01 NEW_VALUE             PIC X(23).
           01 OP                    PIC X(4).

*****************************************************************Program start

PROCEDURE DIVISION USING PART,TOOL,CMS,MATERIAL,THICKNESS,NEW_VALUE,OP.
******************

A1-MAIN SECTION.
A1.
*     Prepare database
      READY CONCURRENT UPDATE.

*     Branch depending on passed OPtion
       EVALUATE OP
         WHEN "TOOL"      PERFORM A2-UPDATE_TOOL
         WHEN "PART"      PERFORM A3-UPDATE_PART
         WHEN "CMS"       PERFORM A4-UPDATE_CMS
         WHEN OTHER       ROLLBACK
      END-EVALUATE.

A1-EXIT.
*     End
       EXIT PROGRAM.

A2-UPDATE_TOOL SECTION.
A2.
*       Only update if different
        IF TOOL NOT = NEW_VALUE
*         Tell user to wait
          CALL "FDV$PUTL" USING BY DESCRIPTOR "SEARCHING... Please wait."
          MOVE NEW_VALUE TO R105_TOOL
          MOVE PART TO R104_PART
          FIND LAST WITHIN CNS105Z USING R105_TOOL
            AT END STORE CNR105
          END-FIND
          FIND LAST WITHIN CNS104Z USING R104_PART RETAINING CNS105A
          RECONNECT CNR104 WITHIN CNS105A
          MOVE TOOL TO R105_TOOL
          FIND LAST WITHIN CNS105Z USING R105_TOOL
          IF NOT (CNS105A MEMBER)
            ERASE CNR105
          END-IF
          COMMIT
        ELSE
          ROLLBACK
        END-IF.
A2-EXIT.
        EXIT.

A3-UPDATE_PART SECTION.
A3.
*       Only update if different
        IF NEW_VALUE NOT = PART
*         Tell user to wait
          CALL "FDV$PUTL" USING BY DESCRIPTOR "SEARCHING... Please wait."
          MOVE CMS TO R100_CMS
          MOVE MATERIAL TO R101_MATERIAL
           MOVE THICKNESS TO R101_THICKNESS
          FIND LAST WITHIN CNS100Z USING R100_CMS
          FIND LAST WITHIN CNS100A USING R101_THICKNESS
                                         R101_MATERIAL
          FIND ALL KEEP-102 WITHIN CNS101A
          MOVE TOOL TO R105_TOOL
          MOVE PART TO R104_PART
          FIND LAST WITHIN CNS105Z USING R105_TOOL
          FIND LAST WITHIN CNS105A USING R104_PART
          MOVE 0 TO EOS
          FIND LAST WITHIN CNS104A
            AT END SET EOS1 TO TRUE
          END-FIND
          PERFORM UNTIL EOS1
            IF CURRENT IS WITHIN KEEP-102
              KEEP CURRENT USING KEEP-VALID-LINK
              FIND PRIOR WITHIN CNS104A
                AT END SET EOS1 TO TRUE
              END-FIND
            END-IF
          END-PERFORM
          MOVE NEW_VALUE TO R104_PART
          FIND LAST WITHIN CNS104Z USING R104_PART RETAINING CNS105A
            AT END STORE CNR104
            NOT AT END RECONNECT CNR104 WITHIN CNS105A
```

```
          END-FIND
          MOVE 0 TO EOS
          FIND LAST WITHIN KEEP-VALID-LINK RETAINING CNS104A
            AT END SET EOS1 TO TRUE
            NOT AT END FREE LAST WITHIN KEEP-VALID-LINK
          END-FIND
          PERFORM UNTIL EOS1
            RECONNECT CNR102 WITHIN CNS104A
            FIND LAST WITHIN KEEP-VALID-LINK RETAINING CNS104A
              AT END SET EOS1 TO TRUE
              NOT AT END FREE LAST WITHIN KEEP-VALID-LINK
            END-FIND
          END-PERFORM
          MOVE PART TO R104_PART
          FIND LAST WITHIN CNS104Z USING R104_PART
          IF NOT (CNS104A MEMBER)
            ERASE
          END-IF
          COMMIT
        ELSE
          ROLLBACK
        END-IF.
A3-EXIT.
        EXIT.

*Update the Company Material Specification record
A4-UPDATE_CMS SECTION.
A4.
*       Only update if different
        IF NEW_VALUE NOT = CMS
*         Tell user to wait
          CALL "FDV$PUTL" USING BY DESCRIPTOR "SEARCHING... Please wait."

*         Set search record
          MOVE CMS TO R100_CMS
          MOVE MATERIAL TO R101_MATERIAL
          MOVE THICKNESS TO R101_THICKNESS
          MOVE TOOL TO R105_TOOL
          MOVE PART TO R104_PART

*         Find record
          FIND LAST WITHIN CNS105Z USING R105_TOOL
          FIND LAST WITHIN CNS105A USING R104_PART
          FIND ALL KEEP-102 WITHIN CNS104A
          FIND LAST WITHIN CNS100Z USING R100_CMS
            AT END ROLLBACK
                   GO TO A4-EXIT
          END-FIND

*         Set not End Of Search
          MOVE 0 TO EOS
          FIND LAST WITHIN CNS100A USING R101-MATERIAL
                                                 R101-THICKNESS

          PERFORM UNTIL EOS1
            IF CURRENT IS WITHIN KEEP-102
              KEEP CURRENT USING KEEP-VALID-LINK
            END-IF
            FIND PRIOR WITHIN CNS101A
              AT END SET EOS1 TO TRUE
            END-FIND
          END-PERFORM
          MOVE NEW_VALUE TO R100_CMS
          FIND LAST WITHIN CNS100Z USING R100_CMS
            AT END MOVE 0 TO CMS-SWITCH
            NOT AT END SET CMS-FOUND TO TRUE
          END-FIND
           IF CMS-FOUND AND NEW_VALUE NOT = "UNKNOWN "
            MOVE 0 TO EOS
            FIND LAST WITHIN CNS100A
            FIND LAST WITHIN KEEP-VALID-LINK RETAINING CNS101A
              AT END SET EOS1 TO TRUE
              NOT AT END FREE LAST WITHIN KEEP-VALID-LINK
            END-FIND
            PERFORM UNTIL EOS1
              RECONNECT CNR102 WITHIN CNS101A
              FIND LAST WITHIN KEEP-VALID-LINK RETAINING CNS101A
                AT END SET EOS1 TO TRUE
                NOT AT END FREE LAST WITHIN KEEP-VALID-LINK
              END-FIND
            END-PERFORM
          ELSE
            STORE CNR100
*           Display FMS form, as pop-up
            CALL "FDV$DISP" USING DESCRIPTOR "CMSFORM"
```

```
*           Prompt user or input
            CALL "FDV$PUTL" USING BY DESCRIPTOR "Enter new material and thickness."
*           Switch on cursor
            CALL "FDV$PUT" USING BY DESCRIPTOR "<ESC>[?25h"
                                   BY DESCRIPTOR "CMSCODE"
*           Show CMS code
            CALL "FDV$PUT" USING BY DESCRIPTOR R100_CMS
                                 BY DESCRIPTOR "CMSCODE"
*           Show current company material code
            CALL "FDV$PUT" USING BY DESCRIPTOR MATERIAL
                                 BY DESCRIPTOR "MAT"
*           Show current material thickness
            CALL "FDV$PUT" USING BY DESCRIPTOR THICKNESS
                                 BY DESCRIPTOR "THICK"
*           Read company material code
            CALL "FDV$GET" USING BY DESCRIPTOR MAT
                                 BY REFERENCE FFT-TRM
                                 BY DESCRIPTOR "MAT"
*           Read material thickness
            CALL "FDV$GET" USING BY DESCRIPTOR THICK
                                 BY REFERENCE FFT-TRM
                                 BY DESCRIPTOR "THICK"
*           Erase FMS form
            CALL "FDV$CLEAR" USING BY VALUE 0
                                   BY VALUE 0

*           Store read data
            MOVE MAT TO R101_MATERIAL
            MOVE THICK TO R101_THICKNESS
            STORE CNR101
              ON ERROR ROLLBACK
                       GO TO A4-EXIT
            END-STORE
            MOVE ZERO TO EOS
            FIND LAST WITHIN KEEP-VALID-LINK RETAINING CNS101A
              AT END SET EOS1 TO TRUE
                 NOT AT END FREE LAST WITHIN KEEP-VALID-LINK
            END-FIND
            PERFORM UNTIL EOS1
              RECONNECT CNR102 WITHIN CNS101A
              FIND LAST WITHIN KEEP-VALID-LINK RETAINING CNS101A
                AT END SET EOS1 TO TRUE
                NOT AT END FREE LAST WITHIN KEEP-VALID-LINK
              END-FIND
            END-PERFORM
          END-IF
          MOVE CMS TO R100_CMS
          FIND LAST WITHIN CNS100Z USING R100_CMS
          FIND LAST WITHIN CNS100A
          FIND LAST WITHIN CNS101A
            AT END FIND CURRENT CNR100
                   ERASE ALL
          END-FIND
          MOVE NEW_VALUE TO CMS
          MOVE MAT TO MATERIAL
          MOVE THICK TO THICKNESS
*         Update database
          COMMIT
        ELSE
*         Routine called with identical code, ignore
          ROLLBACK
        END-IF.
A4-EXIT.
        EXIT.
*****************************************************************End Of Program
```

```
1   iTAPEV ;CMT;tSGV Validate:ITAPE:Patch Tapes;25-SEP-1995;WHITEI;SWIFT
2   REV  ;26-SEP-1995 10:39;WHITEI;1.1;?CMT;IHCS v1.1;901;SGEN compiled
3        ;
4        G START
5   V    ;Insert validation here
6   V    ; Validate Z
7        I zIN<1!(zIN>7) S zER="Please enter a number between 1 and 7"
8   VQ1  Q
9   V2   ; Validate SYS
10       I '$D(^[u,v]iTAPE("SYS",zIN)) S zER="System does not exist (press PF1 for code
list)." Q
11       S CPSYS=$S(zIN="IHCS":"",1+z5N)
12  VQ2  Q
13  V3   ; Validate SITE
14       I zIN'?1.3APN S zER="Please enter the Site's name." Q
15       I '$D(^[u,v]iPAT("SITE",$S(SYS="":"IHCS",1:SYS),zIN)) S zER="Site does not exist
(press PF1 for code list)." Q
16       I ^[u,v]iPAT("SITE",$S(SYS="":"IHCS",1:SYS),zIN)="" S zER="Site details have NOT been
set up (contact CMT for further details)."
17  VQ3  Q
18  V4   ; Validate CONA
19       I zIN'="",zIN'?1.40AP S zER="Invalid Site contact name."
20  VQ4  Q
21  V5   ; Validate SCD
22       I zIN'="",zIN'?3N S zERR="Invalid Short Code."
23  VQ5  Q
24  V6   ; Validate NODE
25  VQ6  Q
26  V7   ; Validate ACC
27       I zIN'==,zIN'?1.0A S zER="Invalid Account name."
28  VQ7  Q
29  V8   ;Validate OPER
30       Q:zIN=""
31       I '$D(^[u,v]iTAPE("OP",zIN)) S zER="system not known (Press PF1 for available
systems)."
32  VQ8  Q
33  V9   ; Validate DSM
34       I '$D(^[u,v]iTAPE("LAN",zIN)) S zER="Language not known (Press PF1 for code list.)"
35  VQ9  Q
36  V10  ; Validate TAPE
37       I '$D(^[u,v]iTAPE("TT",zIN)) S zER="Tape type not known (Press PF1 for code list)."
38  VQ10 Q
39  V11  ; Validate TXTN
40  VQ11 Q
41  V12  ; Validate ACT1
42       I NCH="C",zIN="ad" S zER="You must "_zFL_"A"_zNV_zDI_"ccept or
"_zFL_"R"_zNV_zDI_eject the changes first or e"_zFL_"X"_zNV_zDI_"it." Q
43       I "A^AD^RCX"'[zIN S zER="Not a valid action prompt." Q
44  VQ12 Q
45  V13  ; VALIDATE type
46       I '$D(^[u,v]iTAPE("TT",zIN)) S zER="Tape Type not known (Press PF1 for code list)."
47  VQ13 Q
48  V14  ; Validate NUM
49       I '$D(^[u,v]iTAPE(TYPE)),zIN'="N" S zER="No tapes set up - enter ""N"" to set up a
new Tape." Q
50       S (MX,MX1)="" F  S MX=$O(^[u,v]iTAPE(TYPE,MX)) Q:MX=""  I MX'="" S MX1=MX
51       I zIN="N" S MX1=MX1+1 Q
52       I zIN'?.2N S zER="Invalid Input - Enter Tape number (or ""N"" to set up a new Tape)."
Q
53       I zIN>MX1 S zER="Enter a Tape number between 1 and "_MX1_" (or ""N"" to set up a new
Tape)." Q
54       I '$D(^[u,v]iTAPE(TYPE,zIN)) S zER="Tape no."_zIN_" does not exist."
55  VQ14 Q
56  V15  ; Validate SITE
57       I zIN'?.3AP S zER="Please enter the Site's name." Q
58       I '$D(%[u,v]iPAT("SITE",$S(SYS="":"IHCS",1:SYS),zIN)) S zER="Site details not set
up."
59  VQ15 Q
60  V16  ; Validate VERSION
61       I '$D(u,v]iPAT($S(CPSYS="":"IHCS",1:CPSYS)_"V",zIN)) S zER="Version does not exist
(Press PF1 for code list)"
62  VQ16 Q
63  V17  ; Validate SEQ
64       Q:zIN=""
65       I VERSION="" S zER="Error - Version for this UCI has been entered." Q
66       I '$D(^[u,v]iPAT($S(CPSYS="":"IHCS",1:CPSYS)_"V",VERSION,"PR",zIN)) S zER="No such
Patch release (Press PF1 for code list)."
67  VQ17 Q
68  V18  ; Validate SENT
69       D ^zDVC(zIN)
70  VQ18 Q
71  V19  ; Validate RET
```

```
72       Q:zIN=""
73       D ^zDVC(zIN)
74       I $P(TAPDET,"`",4)>ADATSTO S zER="Hay !!, so the tape was returned before it was sent
?"
75  VQ19 Q
76  V20  ; Validate ACT2
77       I "C`R`X"'[zIN S zER="Not on the list of Action prompts."
78  VQ20 Q
79  V21  ; Validate SITE
80       I zIN'?1.3AP S zER="Please enter the site's name." Q
81       I '$D(^[u,v]iPAT("SITE",$S(CPSYS="":"IHCS",1:CPSYS),zIN)) S zER="Site does not exist
(Press PF1 for code list)." Q
82       I $P(^[u,v]iPAT("SITE",$S(CPSYS="":"IHCS",1:CPSYS),zIN),"`",7)="" S zER="Site details
not set up."
83  VQ21 Q
84  V22  ; Validate NUM
85       I zIN'?.2N S zER="Enter a number between 1 and 99" Q
86       I '$D(^[u,v]iTAPE(TYPE,zIN)) S zER="No such tape." Q
87       I $P(^[u,v]iTAPE(TYPE,zIN),"`",5)=""&($P(^[u,v]iTAPE(TYPE,zIN),"'",4)'="") S
zER="Tape not yet returned."
88  VQ22 Q
89  V23  ; Validate VER
90       I '$D(^[u,v]iPAT($S(CPSYS="":IHCS",1:CPSYS)_"V",zIN)) S zER="Version does not exist
(Press PF1 for code list)."
91  VQ23 Q
92  V24  ; Validate SEQ
93       Q:zIN=""
94       I VER="" S zER="Error - Version for this UCI has been entered." Q
95       I '$D(^[u,v]iPAT($S(CPSYS="":IHCS",1:CPSYS)_"V",VER,"PR",zIN)) S zER="No such Patch
release (Press PF1 for code list)."
96  VQ24 Q
97  V25  ; Validate ACT3
98       I "A`C`X"'[zIN S zER="Not a valid action prompt."
99  VQ25 Q
100 V26  ; Validate UCI
101      I zIN?3U S zIN=$O(^[u,v]iPAT("SITE,SYS,SITE,zIN))
102      I zIN'?3U1","3U S zER="Invalid UCI format (See help for details)."
103 VQ26 Q
104 V27  ; Validate ENV
105 VQ27 Q
106 V28  ; Validate LT
107      I "L`T"'[zIN S zER="Invalid input - Enter 'L' (Live) or 'T' (Training) only."
108 VQ28 Q
109 V29  ; Validate VER
110      Q:zIN=""
111      I '$D(^[u,v]iPAT($S(CPSYS="":IHCS",1:CPSYS)_"V",zIN)) S zER="Version does not exist
(Press PF1 for code list)."
112 VQ29 Q
113 V30  ; Validate SEQ
114      Q:zIN=""
115      I VER="" S zER="Error - Version for this UCI has not been entered." Q
116      I '$D(^[u,v]iPAT($S(CPSYS="":IHCS",1:CPSYS)_"V",VER,"PR",zIN)) S zER="no such Patch
release (Press PF1 for code list)."
117 VQ30 Q
118 V31  ; Validate DU
119      I zIN="" S zER="" Q
120      D ^zDVC(zIN)
121 VQ31 Q
122 V32  ; Validate COM
123 VQ32 Q
124 V33  ; Validate ACT4
125      I "PU`PD[zFK S zIN=zFK
126      I zIN="" S zER="Mandatory field - Refer to help for details." Q
127      I $TR(ACT4(zACTM),"/","`")'[("`"_zIN_"`") S zER="Not on list of action prompts."
128      I NCH="C",zIN="MD" S zER="You must "_zFL_"A"_zNV_zDI_"ccept the changes first or
e"_zFL_"X"_zNV_zDI_"it." Q
129      I zIN="DU",LIN4(1)="" S zER="No UCI's to deselect." Q
130      I "DU`A"[zIN,UCI="" S zER="No UCI selected." Q
131      I zIN="DU",$D(^[u,v]iPAT("SITE",SYS,SITE,UCI)) S zER="UCI "_UCI_" does not already
exist (you didn't press "_zNI_"A"_zDI_"ccept did you?)." Q
132      I "PU`PD"[zIN,PAGES=1 S zER="No other pages to display." Q
133      I zIN="PD",PPN=PAGES S zER="No more pages down (try page up)."
134      I zIN="PU",PPN=1 S zER="No more pages up (try page down)."
135 VQ33 Q
136 VQ   ; End of validation
137 START;
138      D INIT
139 VA   ;
140 VA1  S VLSN=1 Q:SVSTOP'>1 I $E($SWOO,8) S zIN=Z D VMF Q:zER'=""  D V1 Q:zER'=""
141 VA2  S VLSN=2 Q:SVSTOP'>2 I $E($SWOO,9) S zIN=SYS D VMF Q:zER'=""  D V2 Q:zER'=""
142 VA3  S VLSN=3 Q:SVSTOP'>3 I $E($SWOO,10),'$E(sSWIO,10) S zIN=SITE D VMF Q:zER'=""  D V3
Q:zER'=""
143 VA4  S VLSN=4 Q:SVSTOP'>4 I $E($SWOO,12) S zIN=CONA D V4 Q:zER'=""
144 VA5  S VLSN=5 Q:SVSTOP'>5 I $E($SWOO,13) S zIN=SCD D V5 Q:zER'=""
145 VA6  S VLSN=6 Q:SVSTOP'>6 I $E($SWOO,14) S zIN=NODE
```

```
146 VA7  S VLSN=7 Q:SVSTOP'>7 I $E($SWOO,15) S zIN=ACC D V7 Q:zER'=""
147 VA8  S VLSN=8 Q:SVSTOP'>8 I $E($SWOO,16) S zIN=OPER D V8 Q:zER'=""
148 VA9  S VLSN=9 Q:SVSTOP'>9 I $E($SWOO,17) S zIN=DSM D VMF Q:zER'=""  D V9 Q:zER'=""
149 VA10  S VLSN=10 Q:SVSTOP'>10 I $E($SWOO,18) S zIN=TAPE D VMF Q:zER'=""  D V10 Q:zER'=""
150 VA11  S VLSN=11
151 VA12  S VLSN=12 Q:SVSTOP'>12 I $E($SWOO,20) S zIN=ACT1 D VMF Q:zER'=""  D V12 Q:zER'=""
152 VA13  S VLSN=13 Q:SVSTOP'>13 I $E($SWOO,22) S zIN=TYPE D VMF Q:zER'=""  D V13 Q:zER'=""
153 VA14  S VLSN=14 Q:SVSTOP'>14 I $E($SWOO,23) S zIN=NUM D VMF Q:zER'=""  D V14 Q:zER'=""
154 VA15  S VLSN=15 Q:SVSTOP'>15 I $E($SWOO,24) S zIN=SITE D VMF Q:zER'=""  D V15 Q:zER'=""
155 VA16  S VLSN=16 Q:SVSTOP'>16 I $E($SWOO,25) S zIN=VERSION D VMF Q:zER'=""  D V16 Q:zER'=""
156 VA17  S VLSN=17 Q:SVSTOP'>17 I $E($SWOO,26) S zIN=SEQ D VMF Q:zER'=""  D V17 Q:zER'=""
157 VA18  S VLSN=18 Q:SVSTOP'>18 I $E($SWOO,27) S zIN=SENT D VMF Q:zER'=""  D V18 Q:zER'=""
158 VA19  S VLSN=19 Q:SVSTOP'>19 I $E($SWOO,28) S zIN=RET D V19 Q:zER'=""
159 VA20  S VLSN=20 Q:SVSTOP'>20 I $E($SWOO,29) S zIN=ACT2 D VMF Q:zER'=""  D V20 Q:zER'=""
160 VA21  S VLSN=21 Q:SVSTOP'>21 I $E($SWOO,30) S zIN=SITE D VMF Q:zER'=""  D V21 Q:zER'=""
161 VA22  S VLSN=22 Q:SVSTOP'>22 I $E($SWOO,32) S zIN=NUM D VMF Q:zER'=""  D V22 Q:zER'=""
162 VA23  S VLSN=23 Q:SVSTOP'>23 I $E($SWOO,33) S zIN=VER D VMF Q:zER'=""  D V23 Q:zER'=""
163 VA24  S VLSN=24 Q:SVSTOP'>24 I $E($SWOO,34) S zIN=SEQ D VMF Q:zER'=""  D V24 Q:zER'=""
164 VA25  S VLSN=25 Q:SVSTOP'>25 I $E($SWOO,35) S zIN=ACT3 D VMF Q:zER'=""  D V25 Q:zER'=""
165 VA26  S VLSN=26 Q:SVSTOP'>26 I $E($SWOO,36) S zIN=UCI D V26 Q:zER'=""
166 VA27  S VLSN=27 Q:SVSTOP'>27 I $E($SWOO,37) S zIN=ENV
167 VA28  S VLSN=28 Q:SVSTOP'>28 I $E($SWOO,38) S zIN=LT D VMF Q:zER'=""  D V28 Q:zER'=""
168 VA29  S VLSN=29 Q:SVSTOP'>29 I $E($SWOO,39) S zIN=VER D VMF Q:zER'=""  D V29 Q:zER'=""
169 VA30  S VLSN=30 Q:SVSTOP'>30 I $E($SWOO,40) S zIN=SEQ D V30 Q:zER'=""
170 VA31  S VLSN=31 Q:SVSTOP'>31 I $E($SWOO,41) S zIN=DU D V31 Q:zER'=""
171 VA32  S VLSN=32 Q:SVSTOP'>32 I $E($SWOO,42) S zIN=COM
172 VA33  S VLSN=33 Q:SVSTOP'>33 I $E($SWOO,47) S zIN=ACT D V33 Q:zER'=""
173 VAQ   ;
174 EXIT  S VLSN="Z33"
175 EXITQ Q
176 INIT  ;
177       S zRN="iITAPEV"
178 INITQ Q
179 VMF   S:zIN="" zER="Mandatory field - Refer to help details." Q
180 DOC   ; insert any documentation references here
181 insert ; Start and End label insertion points
182       ;VALIDATE;V;V;VQ;V
183       ;INITIALISE;I;INIT;INITQ;;A
184       ;VALFLOW;VA;VA;VAQ;VA
185 Z     S RN=$T(+0) X "ZS @RN" D ^zREVI Q
```

10-DOS batch

```
@ECHO OFF
REM DUP_FILE.BAT
REM by Iain White (c) 1992 BAe (Filton)
REM This program searches for duplicate filenames in two directories.
REM
REM Check parameters
IF "%1" == "" GOTO HELP
REM Make sure directories exist
IF NOT EXIST %1\*.* GOTO ERR1
REM Tell user
ECHO.
ECHO Checking files...
ECHO.
REM Search for duplicates
FOR %%F IN (*.*) DO IF EXIST %1\%%F ECHO %%F is in both places.
ECHO.
ECHO Press a key...
PAUSE > NUL
REM Search for non-duplicates
FOR %%F IN (*.*) DO IF NOT EXIST %1\%%F ECHO %%F is not in %1
ECHO Done.
REM Finish
GOTO END
REM
REM Help
:HELP
ECHO.
ECHO Program name is %0.BAT by Iain White (c) 1992 BAe (Filton)
ECHO.
ECHO This program searches for duplicate filenames in two directories.
ECHO.
ECHO Change to one of the directories to test and
ECHO Type %0 directory
ECHO.
GOTO END
REM
REM Error 1
:ERR1
ECHO Directory %1 does not exist.
GOTO END
REM
REM Finish
:END
```

11-Turbo C

```
/*****************************************************************************/
/* S81 BEATES Windows                                                        */
/* Copyright (c) British Aerospace (Civil Aircraft) Limited                  */
/* Date as last amendment                                                    */
/*****************************************************************************/
/* Name     : Menu.c                                                         */
/* Purpose  : Menu Manager                                                   */
/* Version  : V2.4                                                           */
/* By       : Iain White                                                     */
/* Compiler : Turbo C V2.0                                                   */
/*****************************************************************************/


/* Include files */
#include <stdio.h>                                  /* Standard I/O */
#include <conio.h>                                  /* Console I/O */
#include <stdlib.h>                                 /* Standard library */
#include "keys.h"                                   /* S81 key definitions */
#include "window.h"                                 /* S81 Window Manager */

/* External variables */
extern int VSG;                                     /* Video segment */

/* Function prototypes */
WINDOW *open_menu(char *mnm,MENU *mn, int hsel);
int gethmenu(MENU *mn, WINDOW *hmenu, int hsel);
int getvmn(MENU *mn, WINDOW *hmenu, int *hsel, int vsel);
int haccent(MENU *mn, WINDOW *hmenu, int hsel, int vsel);
void dimension(char *sl[], int *ht, int *wd);
void light(MENU *mn, WINDOW *hmenu, int hsel,int d);

/* Display & process a bar menu */
void menu_select(char *name, MENU *mn)
  {
  WINDOW *open_menu();
  WINDOW *hmenu;
  int sx,sy;
  int hsel = 1,vsel;

  curr_cursor(&sx,&sy);                             /* Store cursor pos */
  cursor(0,26);                                     /* Hide cursor */
  hmenu = open_menu(name,mn,hsel);                  /* Open menu */
  while (hsel = gethmenu(mn,hmenu,hsel))            /* Until horz. selection */
    {
    vsel = 1;                                       /* Fist selection */
    while (vsel = getvmn(mn,hmenu,&hsel,vsel))      /* Until vet. selection */
      {
      delete_window(hmenu);                         /* Remove drop down menu */
      set_help("",0,0);                             /* Kill help pointer */
      (*(mn+hsel-1)->func[vsel-1])(hsel,vsel);
      hmenu = open_menu(name,mn,hsel);              /* Back to horz. menu */
      }
    }
  delete_window(hmenu);                             /* Remove bar menu */
  cursor(sx,sy);                                    /* Restore cursor */
  }

/* Open a horizontal menu */
static WINDOW *open_menu(char *mnm,MENU *mn, int hsel)
  {
  int i = 0;
  WINDOW *hmenu;

  set_help("menu    ",30,10);                       /* Set help */
  hmenu = establish_window(0,0,3,80);               /* Create window */
  set_title(hmenu,mnm);                             /* Set title */
  set_colors(hmenu,ALL,BLUE,AQUA,BRIGHT);           /* Set colours */
  set_colors(hmenu,ACCENT,WHITE,BLACK,DIM);
  set_border(hmenu,0);                              /* No border */
  display_window(hmenu);                            /* Show window */
  while ((mn+i)->mname)
    wprintf(hmenu, " %-10.10s ",(mn+i++)->mname);
  light(mn,hmenu,hsel,1);                           /* High light option */
  cursor(0,26);                                     /* Hide cursor */
  return hmenu;
  }

/* Get a horizontal selection */
static int gethmenu(MENU *mn, WINDOW *hmenu, int hsel)
  {
  int sel;
```

```
  light(mn,hmenu,hsel,1);                               /* High light */
  while (TRUE)
    {
    switch (sel = get_char())                           /* Get key */
      {
      case FWD:
      case BS:   hsel = haccent(mn,hmenu,hsel,sel);     /* Move selection */
                 break;
      case ESC:  return 0;                              /* Quit */
      case '\r': return hsel;                           /* Select */
      default:   putchar(BELL);                         /* Other ring bell */
                 break;
      }
    }
  }

/* Pop down a vertical window */
static int getvmn(MENU *mn, WINDOW *hmenu, int *hsel, int vsel)
  {
  WINDOW *vmenu;
  int ht = 10, wd = 20;
  char **mp;

  while (1)
    {
    dimension((mn+*hsel-1)->mselcs,&ht,&wd);            /* Calculate size */
    vmenu = establish_window(2+(*hsel-1)*12,2,ht,wd);   /* Create window */
    set_colors(vmenu,ALL,BLUE,AQUA,BRIGHT);             /* Set colours */
    set_colors(vmenu,ACCENT,WHITE,BLACK,DIM);
    set_border(vmenu,4);                                /* Set border */
    display_window(vmenu);                              /* Show window */
    mp = (mn+*hsel-1)->mselcs;                          /* Calculate pointer */
    while(*mp)
      wprintf(vmenu,"\n%s",*mp++);                      /* Write options */
    vsel = get_selection(vmenu,vsel,"");                /* Process menu */
    delete_window(vmenu);                               /* Erase drop down */
    if (vsel == FWD || vsel == BS)                      /* Left or right */
      {
      *hsel = haccent(mn,hmenu,*hsel,vsel);             /* Move bar menu select */
      vsel = 1;
      }
    else
      return vsel;
    }
  }

/* Manage the horizontal menu selection accent */
static int haccent(MENU *mn,WINDOW *hmenu,int hsel, int sel)
  {
  switch(sel)                                           /* Select on choice */
    {
    case FWD:                                           /* Forwards */
      light(mn, hmenu,hsel,0);                          /* High light off*/
      if ((mn+hsel)->mname)                             /* Another option */
        hsel++;                                         /* Right */
      else
        hsel = 1;                                       /* First option */
      light(mn,hmenu,hsel,1);                           /* High light */
      break;
    case BS:                                            /* Backwards */
      light(mn,hmenu,hsel,0);                           /* High light off */
      if (hsel == 1)                                    /* Firset option */
        while ((mn+hsel)->mname)                        /* Find last option */
          hsel++;                                       /* Move to last option */
      else
         --hsel;                                        /* Left */
      light(mn,hmenu,hsel,1);                           /* High light */
      break;
    default:                                            /* Ignore */
      break;
    }
  return hsel;
  }

/* Compute a menu's height & width */
static void dimension(char *sl[], int *ht, int *wd)
  {
  unsigned strlen(char *);                              /* String length */

  *ht = *wd = 0;                                        /* Initialise to zero */
  while (sl [*ht])                                      /* For each */
    {
    *wd = max(*wd, strlen(sl [*ht]));                   /* Record longest option */
    (*ht)++;                                            /* Increment horz. count */
```

```
  }
  *ht += 2;                                             /* Add for border */
  *wd += 2;                                             /* Add for border */
  }

/* Accent a horizontal menu selection */
static void light(MENU *mn, WINDOW *hmenu, int hsel, int d)
  {
  if (d)                                                /* Select/deselect flag */
    reverse_video(hmenu);                               /* Reverse video */
  wcursor(hmenu,(hsel-1)*12+2,0);                       /* Position cursor */
  wprintf(hmenu,(mn+hsel-1)->mname);                    /* Write option */
  normal_video(hmenu);                                  /* Rest video */
  cursor(0,26);                                         /* Hide cursor */
  }
```